Caffeモデルを使った画像認識実習課題の開発

吉　武　春　光

西南学院大学商学論集
第67巻　第3・4合併号　抜刷
2021（令和3）年 3 月 発行

Caffeモデルを使った画像認識実習課題の開発

吉　武　春　光

１．まえがき

筆者は、ここ数年、深層学習(Deep Learning)という手法を研究してきた。吉武(2018)において言語データに対して研究を行い、吉武(2019) において画像データに対して研究を行い、深層学習の優位性と問題点を把握した。

最近の深層学習の素晴らしい成果を受けて、政府は AI戦略2019 を行う決定をした。これは、日本が 5G 開発で他国よりも遅れをとったので、国内の ICT 産業の将来を考えての決定らしい。この AI戦略2019 は、AI に結びつくデータサイエンス教育を、文理を問わず全大学生に実施するという内容である。これを受けて、本学でもデータサイエンス教育を全学生に対して実施するというカリキュラム改正が進行中である。

筆者は、データサイエンス教育の一貫として、公開されている学習済みモデルを利用して学生が画像認識を試みることができるプログラムを開発した。本稿では、その概要を報告する。

２．深層学習の学習済みモデル

本章では、深層学習の学習済みモデルについて述べる。深層学習には学習と推論という２つの段階がある。学習済みモデルとは、学習が終わった段階で生成されたモデルのことを指す。深層学習の学習には多大な量のデータが必要であるので、学習済みモデルを再利用できれば、深層学習がやりやすくなる。

2.1 Caffe と Model Zoo

吉武(2019)で述べたようにディープラーニング用のフレームワークは色々とあるが、公開された学習済みモデルを活用するとなると、便利なのが Caffe である。Caffe はカリフォルニア大学で開発されたディープラーニング用のフレームワークである。ディープラーニング開発の初期に公開されたために、数多くの利用があり、Model Zoo というサイトに数多くの学習済みモデルが公開されている。そこで、本研究では、学習済み Caffe モデルをフレームワーク Chainer に読み込ませて再利用する場合を試すことにした。

2.1.2 Model Zoo からのモデルの取得

Model Zoo には多くの学習済みモデルが公開されているが、今回は、分かりやすいモデルとして bvlc_googlenet を使用した。bvlc_googlenet は1000カテゴリー(種類)の画像分類を行うモデルです。画像サイズは 224 × 224 ピクセルである。

2.1.3 Chainer における Caffe モデルの利用

筆者が吉武春光(2019)で用いたディープラーニング用のフレームワーク Chainer には、学習済み Caffe モデルを利用する仕組みが備わっている。そこで、今回の研究では、Chainer を使って 学習済み Caffe モデルを利用することにした。新納 (2016) が参考になった。

3．画像識別課題のためのプログラム

本研究では、Web 画面から画像ファイルを指定し、その画像ファイルを認識分類し、得られた結果の上位10個を表示するようにした。使用したプログラムは Python用フレームワーク Flask を使用しPythonを使って記述した。

なお、筆者の Ubuntu Server 18.04 上で Flask を動かした。GPU として性能 11.34TFLOPS の NVIDIA GTX 1080 Ti を搭載している。

全体の流れは図１の通りとなる。

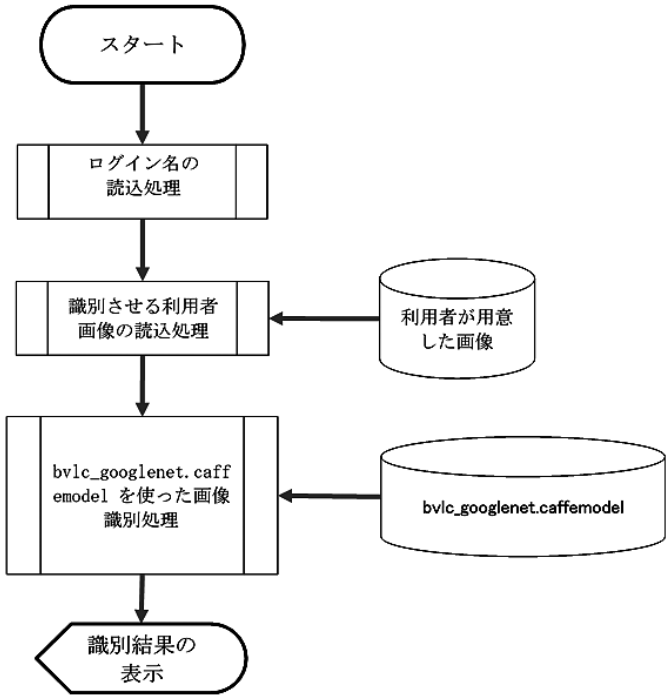

図１　画像識別処理の流れ

3.1 Flaskプログラムの設計

付録に載せたように、メインのプログラム upload.py を作成し、更にWeb画面用templateとして次の３つを設計した。

表１　画面設計

画面の目的	template名
ログイン名の読み込み用	read_uid.html
画像を指定して読み込ませる用	upload_img.html
識別結果表示用	result.html

3.2 Flaskをhttpsでアクセスできるようにする

本研究では、学生に学外からでもアクセスできるようにSSLを使うことが必要になる。ブラウザ上でPythonへの読み書きをする仕組みとしては、吉武(2019)で使用したJupyterノートブックにおいてSSL supportをenableにする方法があるが、Jupyterノートブックは研究開発用であり、アプリケーションの形態として利用者に使わせるには向いていない。そこで、今回は、Flask自身をSSL対応する方法やWebサーバで対応する方法などを試した。

3.2.1 Flask自身をSSL対応する方法

Flaskの設定を調べたら、SSLの秘密鍵と公開鍵を指定することによりFlask内臓のWebサーバをSSL対応することが可能と分かったmotojapan(2017)。しかし、SSLのポート番号をサーバ標準の443にすることができないことが判明した。本研究で使用するサーバは学内に置いており、セキュリティの観点で外部から学内へ通過できるポート番号は443に限定されているので、Flask自身をSSL対応する方法を諦めることにした。

3.2.2 WSGI対応Webサーバで対応する方法

まずWSGI(ウィズギー)はWeb Server Gateway Interfaceの頭文字をとったもので、WebサーバとWebアプリケーションフレームワークをつなぐ共通の汎用的に使えるインターフェースのことである(保坂2008)。背景としては、元々Webの仕組みはPHPとHTMLを前提としたものであったので、他のプログラミング言語で書かれたソフトウェアをWebブラウザ上で動かす（入出力させる）ためには、他のプログラミング言語ごとに独自の実装がされていた。今回の研究で使用したプログラミング言語Pythonにおいても幾つもの実装が存在していた。この問題を解決するために考案されたのがWSGIである。WSGIはPythonのAPIとして定義されており、実装されたものは次の２種類がある。

・Pythonで書かれた専用サーバGunicornやuWSGI

・汎用 Web サーバ Apache の追加モジュールとして開発された mod_wsgi
なお、この WSGI に着想を得て、他のプログラミング言語でも、同様のインターフェースが開発されている。例えば、プログラミング言語 Perl 用に PSGI (Perl Web Server Gateway Interface)、プログラミング言語 Ruby 用に Rack (Ruby Web Server Interface) がある。

3.2.3 uWSGI

本研究では、まずPythonで書かれた専用サーバuWSGIを試してみた。筆者が使用している Ubuntu Server では uWSGI のインストールは簡単であり pip3 install uwsgi コマンドを入力するだけであった。uWSGI を使って Pythonプログラムを実行させる際のコマンドの指定は次の通りである。

```
$ sudo uwsgi --master --https 160.23.145.22:443,hy-svr.cer,hy-svr.key -w upload:app
```

ここで、学外からのアクセスを許すために SSL 証明書を指定する必要があった。それが hy-svr.cer と hy-svr.key である。ここで 160.23.145.22 は実験に使用したサーバ hy-svr.seinan-gu.ac.jp の IP アドレスである。

実際に動かしてみたら、安定に動作するのであるが、問題点として浮かび上がったの、ポート番号 443 を専有してしまうので他の https アプリケーションを動かすことができない」ということである。実験に使用したサーバは他の Web アプリケーションも動かす必要があったので、uWSGI を使うことは諦めた。

3.2.4 mod_wsgi

次に、汎用 Web サーバ Apache の追加モジュールとして開発された mod_wsgi を試すことにした。振り返ってみると簡単であったが、乗り越えるべき点が多かった。

まず、インストールは Ubuntu の apt コマンドを使うと簡単であるが、mod_wsgi のバージョンが古い可能性がある。そこで、pip コマンドを使って最新版の mod_wsgi をインストールすることにした。

```
$ sudo pip3 install mod_wsgi
```

上手くインストールできたかどうかは、mod-wsgiの簡易Webサーバーを起動することで確認できる。次のコマンドを入力する。

```
$ mod_wsgi-express start-server
```

つぎに、mod_wsgi-express コマンドを使って、Apache2のモジュールフォルダにmod_wsgiのモジュールをインストールした。

```
$ sudo mod_wsgi-express install-module
LoadModule wsgi_module "/usr/lib/apache2/modules/mod_wsgi-py36.cpython-36m-x86_64-linux-gnu.so"
WSGIPythonHome "/usr"
```

この結果は /etc/apache2/mods-enabled/ の下に wsgi.conf と wsgi.load が上手くリンクされているかどうかで確認できる。
Apache で設定する値の確認は次のコマンドで行える。

```
$ sudo mod_wsgi-express module-config
LoadModule wsgi_module "/usr/local/lib/python3.6/dist-packages/mod_wsgi/server/mod_wsgi-py36.cpython-36m-x86_64-linux-gnu.so"
WSGIPythonHome "/usr"
```

ここで、/usr/lib/apache2/modules/ 下の .so ファイルと /usr/local/lib/python3.6/dist-packages/mod_wsgi/ 下の .so ファイルは同じものである。Python3 が認識する .so ファイルと Apache が認識する .so ファイルが別の場所に置かれているということである。

Apache でアクセスを可能にするためには、Flask プログラムに加えて .wsgi ファイルを作成する必要があった。今回、作成した Flask プログラムは upload.py というファイ名にしたので、upload.wsgi ファイルを upload.py と同じ階層に作成した。upload.wsgi ファイルの中身は次の通り：

```
import sys, os

import logging
# apacheのログに出すために必要
logging.basicConfig(stream = sys.stderr)

sys.path.insert(0, os.path.abspath(os.path.dirname(__file__)))

from upload import app as application
```

そして、次にWebサーバApacheのVirtual Hostにおいて次の設定をした。

```
<Directory "/var/www/kougi/">
    Require all granted
</Directory>
Alias /kougi "/var/www/kougi"

WSGIDaemonProcess www-data user=www-data group=www-data
    python-home=/usr python-path=/var/www/kougi:/usr/lib/python3.6/site-packages
WSGIProcessGroup www-data
WSGIScriptAlias /ai /var/www/kougi/upload.wsgi
```

４．課題の実施

筆者が行っている「知識情報処理論」という科目で、ディープ・ラーニングを使った画像認識を経験し考察を加えるという課題を出した。

最初の画面で、受講生のログイン名を入力してもらい、次の画面にて、読み込ませる画像を指定して、実際に識別をさせた。次の図２に、その識別の様子を示す。

図２　課題プログラムの処理の様子

mod_wsgi を使った Apache は安定した動作を行っており、合計104名の受講生が、この課題を行ったが、問題は発生しなかった。

５．あとがき

本研究では、データサイエンス教育の一貫として、学生が画像認識を試みることができるプログラムを開発し、実習課題として使用した。フレームワーク Chainer を使って、学習済み Caffe モデルを利用することで実現した。

今回の研究では、学習済み Caffe モデルをそのまま使用したが、分類カテゴリーが異なる場合などには、ファインチューニングや転移学習と呼ばれるモデルの再学習を行うと良いらしい。今後は、ファインチューニングや転移学習にも取り組んで行く。

なお、深層学習フレームワーク Chainer は開発が終了し、今後は別の深層学習フレームワーク PyTorch に統合されるという発表があった。筆者の今後の研究は PyTorch を使うことになるであろう。

参考文献

@lamplus(2018), "ラズベリーパイにApacheを導入し、pythonファイルを実行するまでの備忘録", https://qiita.com/lamplus/items/9877849d3108e2c6d0df, 2021.1.10アクセス

bvlc_googlenet, https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet, 2020.9.10アクセス

Caffe install, https://caffe.berkeleyvision.org/install_apt.html, 2020.11.30アクセス

Caffe Model Zoo, https://github.com/BVLC/caffe/wiki/Model-Zoo,
2020.11.30アクセス

The uWSGI project, https://uwsgi-docs.readthedocs.io/en/latest/,
2020.12.25アクセス

uWSGI入門, https://www.python.ambitious-engineer.com/archives/1959,
2020.12.25アクセス

Munesada, Yohei(2017), "[Python] mod_wsgiを使ってPython3.6をApacheで動かす（CentOS6系）", https://www.yoheim.net/blog.php?q=20170206,
2021.1.10アクセス

mod_wsgi, https://modwsgi.readthedocs.io/en/develop/index.html,
2021.1.10アクセス

"mod_wsgi (Apache)", Flask 公式解説, https://flask.palletsprojects.com/en/1.1.x/deploying/mod_wsgi/,
2021.1.10アクセス

motojapan(2017), "Webサーバーをhttps対応する方法", http://motojapan.hateblo.jp/entry/2017/12/14/083635,
2021.1.10アクセス

wsgi-expressコマンド解説, "https://pypi.org/project/mod-wsgi/",
2021.1.10アクセス

新納浩幸(2016),"Chainerによる実践深層学習", オーム社

保坂翔馬(2008),"WSGIとPythonでスマートなWebアプリケーション開発を", https://gihyo.jp/dev/feature/01/wsgi/0001,2019.1.10アクセス

吉武春光(2018): "学生の提出レポート解析に文脈ベクトルを使う", 西南学院大学商学論集, Vol. 64, No. 4, pp. 79-95, 2018.3月

吉武春光(2019): "深層学習を用いた学生の受講態度の推定", 西南学院大学商学論集, Vol. 65, No. 4, pp. 215-236, 2019.3月

付録

図3 作成したPythonフレームワークFlaskのプログラムupload.py(1/3)。

```
import os
#request フォームから送信した情報を扱うためのモジュール
#redirect ページの移動
#url_for アドレス遷移
from flask import Flask, render_template, request, redirect, url_for, flash
#ファイル名をチェックする関数
from werkzeug.utils import secure_filename
#画像のダウンロード
from flask import send_from_directory

#画像のアップロード先のディレクトリ
UPLOAD_FOLDER = '/var/www/kougi/uploads'
upload_folder_name = 'uploads'
#アップロードされる拡張子の制限
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'gif'])

app = Flask(__name__)

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allwed_file(filename):
    #.があるかどうかのチェックと、拡張子の確認
    #OKなら1、だめなら0
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

#ログイン名を入力してもらう
@app.route('/', methods=['GET', 'POST'])
def index():
    global uid
    uid=""
    #read_uid.htmlから/goto_uploads_fileに飛ばす
    return render_template("read_uid.html",uid=uid)

#ファイルを受け取る方法の指定
@app.route('/goto_uploads_file', methods=['GET', 'POST'])
def goto_uploads_file():
  global uid
  #リクエストがポストかどうかの判別
  uid = request.form.get('uid')
  return render_template('upload_img.html', uid=uid)
```

図 4 作成したPython フレームワークFlaskのプログラムupload.py(2/3)。

```
@app.route('/uploads_file', methods=['GET', 'POST'])
def uploads_file():
    global uid
    #データの取り出し
    file = request.files['file']
    #filenameの先頭にログイン名を挿入する
    filename=file.filename
    filename=uid + "-"+ filename
    #ファイルがなかった場合の処理
    if 'file' not in request.files:
      flash('file not in request.files')
      return redirect(request.url)
    #ファイル名がなかった時の処理
    if file.filename == '':
      flash('ファイルがありません')
      return redirect(request.url)
    #ファイルのチェック
    if file and allwed_file(file.filename):
        #危険な文字を削除（サニタイズ処理）
        filename = secure_filename(filename)
        #ファイルの保存
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
#       file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        #アップロード後のページに転送
        return redirect(url_for('uploaded_file', filename=filename))

@app.route('/uploads/<filename>')
#ファイルを表示する
def uploaded_file(filename):
    import numpy as np
    import chainer
    from chainer import cuda, Function, gradient_check, Variable
    from chainer import optimizers, serializers, utils
    from chainer import Link, Chain, ChainList
    import chainer.functions as F
    import chainer.links as L
    from chainer.links import caffe
    from PIL import Image

    imagepath=app.config['UPLOAD_FOLDER']+"/"+filename
    image = Image.open(imagepath).convert('RGB')
    fixed_w, fixed_h = 224, 224
    w, h = image.size
    if w > h:
        shape = (int(fixed_w * w / h), fixed_h)
    else:
        shape = (fixed_w, int(fixed_h * h / w))

    left = (shape[0] - fixed_w) / 2
    top = (shape[1] - fixed_h) / 2
    right = left + fixed_w
    bottom = top + fixed_h
```

図5 作成したPythonフレームワークFlaskのプログラムupload.py(3/3)。

```
    image = image.resize(shape)
    image = image.crop((left, top, right, bottom))
    x_data = np.asarray(image).astype(np.float32)
    x_data = x_data.transpose(2,0,1)
    x_data = x_data[::-1,:,:]

    mean_image = np.zeros(3*224*224).reshape(3, 224, 224).astype(np.float32)
    mean_image[0] = 103.0
    mean_image[1] = 117.0
    mean_image[2] = 123.0

    x_data -= mean_image
    x_data = np.array([ x_data ])


    x = chainer.Variable(x_data)
    func = caffe.CaffeFunction('/var/www/kougi/bvlc_googlenet.caffemodel')
    with chainer.using_config('train', False):
        y, = func(inputs={'data': x}, outputs=['loss3/classifier'])

    prediction = F.softmax(y)
    #ラベルを読み込む
    categories = np.loadtxt('/var/www/kougi/labels.txt', str, delimiter="\n")

    # スコアとラベルを紐づけスコアの高い順にソートする
    result = zip(prediction.data.reshape((prediction.data.size,)), categories)
    result = sorted(result, reverse=True)

    # 上位10個の結果を表示する
    top10=enumerate(result[:10])
    return render_template('result.html', top10=top10, imagepath=imagepath,
upload_folder_name=upload_folder_name, filename=filename)
    #from flask import flash
    #for i, (score, label) in enumerate(result[:10]):
    #        flash ('{:>3d} {:>6.2f}% {}'.format(i + 1, score * 100, label))

    ## prob = F.softmax(y)
    ## labels = open('labels.txt').read().split('\n')
    ## maxid = np.argmax(prob.data[0])
    ## print (labels[maxid], prob.data[0,maxid])

    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)

if __name__ == '__main__':
    app.debug = True #デバッグモード有効化
    app.run(host='160.23.145.22', port=8888)
    #app.run(host='localhost', port=8888)
```

図6 作成したPythonフレームワークFlaskのtemplate read_uid.html

```
<!doctype html>
<html>
  <head>
    <title>ログイン名の入力</title>
  </head>
  <body>
   ログイン名を入力してください：<br/>
                    <form action="/ai/goto_uploads_file" method="post" class="form-inline">
                     ログイン名：<input name="uid" value="{{uid}}" type="text" /><br/>
                     <button type="submit" class="btn btn-default">送信する</button>
                    </form>
  </body>
</html>
```

図7 作成したPythonフレームワークFlaskのtemplate upload_img.html

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>ファイルをアップロードして判定しよう</title>
  </head>
  <body>
    <h2>左の [参照...] で画像を指定してから、右の [Upload] をクリックしてください：<br>
    結果が表示されるまで10秒ほどかかります。</h2>
    <form action="/ai/uploads_file" method="post" enctype="multipart/form-data">
    <p><input type="file" name="file">
    <input type="submit" value="Upload">
    </form>
  </body>
</html>
```

図8 作成したPythonフレームワークFlaskのtemplate result.html

```
<!doctype html>
<html>
  <head>
    <title>{{imagepath}}</title>
  </head>
  <body>
    <img src="/ai/static/{{filename}}" width="400"><br/>
    画像識別結果<br/>
    順位--> 確率--> 識別結果としての種類<br/>
                                        {% for one in top10 %}
                                        {%- set i = one[0] + 1 -%}
                                        {{i}}-->{{one[1][0]}}-->{{one[1][1]}}<br/>
                                        {% endfor %}<br/>
                    <form action="/ai" method="get" class="form-inline">
                     <button type="submit" class="btn btn-default">最初に戻って別の画像を識別させる</button>
                    </form>
  </body>
</html>
{% extends "layout.html" %}
{% block content %}
 <!-- Form
    ================================================== -->
 <div class="form">
  <div class="container">
   <div class="row">
    <div class="col-md-12">
     <p class="lead">
      処理対象ベクトル：<br/>
      ・単位={{target}}<br/>
      ・期間={{year}}{{term}}<br/>
      ・file={{dic}}<br/><br/>
      {% if posword %}
      positive_word={{posword}}<br/>
      {% endif %}
      {% if negword %}
      negative_word={{negword}}<br/>
      {% endif %}
      <br/>
      {% if result %}
      similarity=<br/>
      {% for one in result %}
      {{one}}<br/>
```

```
    {% endfor %}
    {% endif %}
    </p>
    <form action="/wordsel" method="post" class="form-inline">
     positive word: <input name="posword1" value="" type="text" /><input name="posword2" value="" type="text" /><input
name="posword3" value="" type="text" /><input name="posword4" value="" type="text" /><input name="posword5"
value="" type="text" /><input name="posword6" value="" type="text" /><input name="posword7" value="" type="text" />
<br/>
     negative word: <input name="negword1" value="" type="text" /><input name="negword2" value="" type="text" /><input
name="negword3" value="" type="text" /><input name="negword4" value="" type="text" /><input name="negword5"
value="" type="text" /><br/><br/>
    <button type="submit" class="btn btn-default">most_similarを計算する
</button>
    </form>
    <br/>
    <br/>
    <br/>
    <form action="/" method="get" class="form-inline">
    <button type="submit" class="btn btn-default">処理対象を選び直す</button>
    </form>


   </div>
  </div>
 </div>
</div>
{% endblock %}
```